The Experiment Data Depot: a web-based software tool for biological experimental data storage, sharing, and visualization.

William C. Morrell,[‡,§,∇] Garrett W. Birkel,[‡,∥,⊥,∇] Mark Forrer,[‡,§,∥] Teresa Lopez,[‡,§,∥] Tyler W. H. Backman,[‡,∥,⊥] Michael Dussault,[‡] Christopher J. Petzold,[‡,∥,⊥] Edward E. K. Baidoo,[‡,∥,⊥] Zak Costello,[‡,∥,⊥] David Ando,[‡,⊥] Jorge Alonso-Gutierrez,[‡,⊥] Kevin W. George,[‡,⊥] Aindrila Mukhopadhyay,[‡,⊥] Ian Vaino,[‡] Jay D. Keasling,[‡,⊥,¶,∥,⊥] Paul D. Adams,[‡,∥,#] Nathan J. Hillson,[*,‡,∥,⊥,@] and Hector Garcia Martin[*,‡,∥,⊥,△]

‡*DOE Joint BioEnergy Institute, Emeryville, CA, USA.*

§*Biotechnology and Bioengineering and Biomass Science and Conversion Department, Sandia National Laboratories, Livermore, CA, USA.*

∥*DOE Agile BioFoundry, Emeryville, CA, USA.*

⊥*Biological Systems and Engineering Division, Lawrence Berkeley National Laboratory, Berkeley, CA, USA.*

¶*Department of Chemical and Biomolecular Engineering, University of California, Berkeley, CA, USA.*

∥*Department of Bioengineering, University of California, Berkeley, CA, USA.*

⊥*Novo Nordisk Foundation Center for Biosustainability, Technical University Denmark, DK2970-Horsholm, Denmark.*

#*Molecular Biophysics and Integrated Bioimaging Division, Lawrence Berkeley National Laboratory, Berkeley, CA, USA.*

@*DNA Synthesis Science Program, DOE Joint Genome Institute, Walnut Creek, CA.*

△*BCAM, Basque Center for Applied Mathematics, Bilbao, Spain.*

∇*These authors contributed equally.*

E-mail: nhillson@lbl.gov; hgmartin@lbl.gov

Running header

The Experiment Data Depot

Abstract

Although recent advances in synthetic biology allow us to produce biological designs more efficiently than ever, our ability to predict the end result of these designs is still nascent. Predictive models require large amounts of high-quality data to be parametrized and tested, which are not generally available. Here, we present the Experiment Data Depot (EDD), an online tool designed as a repository of experimental data and metadata. EDD provides a convenient way to upload a variety of data types, visualize these data, and export them in a standardized fashion for use with predictive algorithms. In this paper, we describe EDD and showcase its utility for three different use cases: storage of characterized synthetic biology parts, leveraging proteomics data to improve biofuel yield, and the use of extracellular metabolite concentrations to predict intracellular metabolic fluxes.

Keywords: Database, -omics data, data standards, data mining, flux analysis, synthetic biology

The field of biology has undergone a radical transformation in the 20th and 21st centuries: whereas biology had previously been a descriptive science, focused on classifying and explaining biological behavior, the advent of genetic engineering and synthetic biology provides the possibility of changing the instruction set of biological entities and modifying their behavior.[1] The ensuing anticipated industrialization of biology in the 21st century[2] is expected to significantly impact society in several ways: a biobased economy has the potential to address key environmental challenges, transform manufacturing processes, increase the productivity and scope of the agricultural sector, reduce the economy's dependence on oil, improve human health, and grow new jobs and industries.[3]

However, while our capability to create new biological designs is advancing quickly, our ability to predict the outcome of engineered biological systems remains nascent. DNA synthe-

sis productivity improves as fast as Moore's law,[4] and new tools for facile genome engineering have revolutionized our capabilities to introduce site-specific modifications in the genomes of cells and organisms.[5] Nonetheless, while it is increasingly more manageable to make the DNA changes we intend, the end result on cell biology is generally unforeseen.[6]

One of the main obstacles in predicting the behavior of biological systems is a concerning lack of repeatability in bioengineering, as compared to other engineering disciplines. While it is possible to produce a blueprint and specific instructions to construct (*e.g.*) a cell phone in China that will satisfy the same specifications as the same phone built in the U.S., the same is not the case for bioengineered systems.[7] Recent studies by Amgen and Bayer were able to reproduce only 10-30% of biotech findings published in top-tiered journals,[6,8,9] and there is a growing concern regarding lack of reproducibility.[10] This lack of reproducibility not only hampers predictability, but also significantly limits investment in the field: the rule of thumb that has been reported to be applied among venture capitalists is that 50% of studies in top-journals are irreproducible.[8]

Greater predictability and reproducibility requires efficient data, metadata, and protocol collection and sharing.[7] New computational biology approaches for predicting biological behavior are becoming available, ranging from machine learning techniques to mechanistic models.[11–14] However, the large amounts of standardized high-quality data that are needed to rigorously validate or improve these models are lacking. Concurrently, the post-genomic revolution has provided experimentalists with large-scale data sets of -omics data that are orders of magnitude larger than they are typically trained to analyze. Hence, the collaboration between experimentalists and computational specialists could become much more fruitful and frequent through a more robust exchange of data. In the field of synthetic biology, for example, it has been shown that careful characterization of synthetic biological parts enables accurate prediction of full pathway behavior.[15]

However, description of the experimental details is typically only reported in the materials and methods of papers in non-standard, often incomplete, ways.[6] New frameworks such as

the ISA (Investigation/Study/Assay) software[16] are appearing which provide a standardized description of experiment design and metadata that have become the standard way to report results to journals like Nature Scientific Data. This framework and others[17,18] have facilitated the appearance of tools for sharing transcriptomics,[19,20] proteomics,[21–23] metabolomics data,[24] and even combinations of different -omics data types.[25] In parallel, data collection and storage systems based on standards developed for medical purposes (DICOM, Digital Imaging and COmmunication in Medicine[26]) are being applied to synthetic biology part characterization (DICOM-SB[27]). However, none of these tools provides a single data repository for all -omics data types that is able to extract data straight from instrument output, visualize this data, and export the data in formats that are readily applicable to modeling tools and libraries.

Here, we present the Experiment Data Depot (EDD), an online tool designed as a repository of experimental data and metadata (Fig. 1). EDD can uptake experimental data, provide visualization of these data, and produce downloadable data in several standard output formats. The input of data to EDD is performed through automated data streams: each of these input streams automatically parses the standard outputs of the instruments most commonly used for bioengineering. New input streams can be easily added to adapt to local data production. The current version of EDD handles transcriptomics, proteomics, metabolomics, HPLC, and Biolector® fermentation data. EDD provides a quick visualization of imported data that allows for a quality check by showing whether the imported data are within the expected range or not. Since data are stored internally in a relational database, all data output is consistent. Outputs can be provided in terms of different standardized files (Systems Biology Markup Language, SBML,[28,29] or CSV) or through a representational state transfer (RESTful) Application Programming Interface (API, in development). Since the most common complaint of data scientists[30] is that they spend most of their time preparing data for analysis rather than doing the analysis itself, the ability to obtain data in standardized formats should be of great utility. SBML and CSV files can be used in conjunc-

tion with libraries such as COBRApy[31] or Scikit-learn[32] to generate actionable results for metabolic engineering. We showcase this capability by using HPLC data to predict internal metabolic fluxes of cells, and by leveraging proteomic data to improve biofuel yield. We also demonstrate EDD's capability to store information on characterized synthetic biology parts.

EDD is not a LIMS (Laboratory Information Management System): it is not meant to store raw data (*e.g.*, mass spectrometry traces). Rather, it only stores processed *biologically interpretable* data (*e.g.*, metabolite concentrations, protein expression levels, oxygen input rates, *etc.*), *i.e.* data that can be immediately interpreted by a biologist without requiring detailed knowledge of the analytical measurement technique.

Methods

Experiment description terms (EDD ontology)

EDD describes experiments in terms of studies, lines, strains, protocols, assays, measurements, and values (see Fig. 2 for an illustrative example).

- **Study** is used to describe a single continuous experiment meant to answer a single question. For example, an experiment characterizing the properties of a library of promoters in *Escherichia coli* would be a Study. Another example would be screening a panel of mutant enzymes for specificity to a molecule of interest.

- **Line** describes a single culture or line of enquiry within a Study. A single flask with a *E. coli* strain culture, or a well of *Saccharomyces cerevisiae* in a plate, are examples of Lines in EDD. Lines are grouped together under a Study in the EDD hierarchy, therefore a Study contains a set of Lines.

- **Strain** describes the biological entity used in a Line. A Line entry includes information about the strain or enzyme being used, making it possible to search for any Line or Study that uses a specific strain. Multiple Lines within a Study can use the same Strain,

either as biological replicates, or under differing conditions. Additional information concerning the strain(s) and/or plasmid(s) used in a Line is made available through links to the Inventory of Composable Elements (ICE),[33] which serves as a repository for DNA sequences, the physical location in the laboratory freezer, and other strain metadata.

- **Protocol** denotes the method used to obtain information from a Line (*e.g.*, proteomics). A Protocol is not tied to any particular Study; it is any repeatable process meant to be used across many Studies. The description of a Protocol can be anything from a simple list of written instructions, to a reference to a document or manual, or a robot program.

- **Assay** is the application of a Protocol on a specific Line (*e.g.*, using proteomics to study protein expression of Line C1, an *E. coli* culture). Assays are grouped under Lines in the EDD hierarchy, thus a Line contains a set of Assays (see C1-PROT-1 and C1-PROT-2 in Fig. 2).

- **Measurement** describes a quantity measured by an Assay (*e.g.*, the count of phosphoglucose isomerase proteins per cell found using the proteomics protocol on line C1). Some Protocols will measure only one quantity (*e.g.*, optical density at 600 nm), while others could measure multiple quantities (*e.g.*, several proteins for proteomics or several extracellular metabolites for HPLC).

- **Values** are individual points of data for a Measurement. A Measurement could contain only a single value, or several of them.

Key capabilities

Data input

Data input into EDD has been streamlined (Fig. 3 and Screencast 1 in Supporting Information). The data input menu consists of a set of prescribed import modules, plus a more general import option (Fig. 4). The assumption in this design is that typically the same types of data are imported, and new data types are only rarely added. The prescribed data inputs include options for HPLC data, targeted proteomics data, metabolomics concentration data, metabolite labeling patterns (such as those used in ^{13}C Metabolic Flux Analysis[34,35]), transcriptomics data, and data obtained from the m2p-labs Biolector® automated fermentation platform.[36] A specific input format is expected for each data type depending on the data source (e.g. HPLC or Biolector) to standardize and facilitate data input. An example of the data format is shown in the input form as a guidance (see Screencast 1). New data type inputs can be easily added by including a new import module conforming to the interface for import/export modules (see Supporting Information).

Data lacking a specified format or type can be uploaded through a general import option. This option attempts to allow greater flexibility in defining rows and columns of an input table. A large variety of spreadsheet layouts may be handled by the general import, but this requires the user of EDD to define mappings of spreadsheet rows and columns to EDD datatypes.

Visualization

EDD provides visualization of experimental data through interactive tables and graphs (see Fig. 5 and Screencast 2 in Supporting Information). The guiding principle of visualization in EDD is that it is not meant to solve all visualization needs, but rather provide a general overview of datasets via visualization of the most common needs, while the rest can be tackled through data downloads and more sophisticated visualization tools (*e.g.*, Spotfire[37]

or Plot.ly).

The EDD study detail view contains several sections to present different facets of data contained in the study: an overview part ("Overview"), a table describing lines and metadata ("Experiment Description") and an interactive graph displaying all collected data ("Data"). The "Data" section (Fig. 5) allows the user to see different measurements for each line (*e.g.*, acetate concentration for *E. coli* wild type strain or the number of copies of fumC protein in engineered strain p3BB4) via different graph types: line, or bar graphs where data is grouped by varying criteria. An interactive menu allows the user to toggle among different data types or lines, in order to compare them. In this way, one can, for example, compare glucose consumption for several strains, or lactate vs acetate production of a single strain. This visualization gives the researcher a quick data quality check by testing whether the gathered data matches intuitive expectations. The toggling is enabled through progressive filtering of metadata criteria: Line, Strain, Protocol, Assay, Measurement (plus other metadata customized for the Study). The filtering draws one column for each metadata type that has more than one unique value in the Study, then lists the unique values in the column. When a value in the column is checked the overview plot is updated to show only the records related to the checked value. Also, the contents of all the columns to the right of the modified column are updated to show which values remain in the currently visibly subset of records. In this way, the user can progressively drill down into arbitrary groups of their data efficiently (see Screencast 2 in Supporting Information for a demonstration).

The "Experiment Description" section of the Study detail view collects the metadata and descriptors of Lines into a searchable, filterable, and sortable table. Lines can be searched through a box which filters out all lines not meeting the search criteria, and sorted by clicking on headers, as in spreadsheets. The relevant metadata fields can be shown or hidden through an options menu.

Data standardization

EDD provides a single repository of data and a set of unified workflows for data input which facilitate standardized data collection and storage. This standardization facilitates comparison of experiments accumulated over time and provides a unified input for data analysis. Furthermore, detailed protocols and metadata parameters for each type of measurement are stored within EDD. Including this additional context in data standards is important, so the researcher analyzing the data does not need to be the same individual or team who executed the experiments. This decoupling enables effective division of labor and helps improve productivity.[7]

EDD uses PubChem Compound Identifiers (cids) as the primary identifier for tracking metabolites.[38] Common genome-scale models are supported by a pre-generated mapping that connects BiGG[39] identifiers to cids by using ChEBI[40] as an intermediate, as there are BiGG<->ChEBI and ChEBI<->PubChem cross-references, but no direct BiGG<->PubChem cross-references available. For databases other than BiGG, identifier mappings are not automatically resolved to PubChem cids. Novel metabolites not yet included in PubChem can be added to the database via the administration interface, which stores chemical structures as a SMILES[41] string.

Proteins are tracked using the UniProt unique identifier (UPI,[42]), and *E. coli* genes are currently tracked using Blattner numbers (b-numbers[43]). Support for NCBI GenBank[44] accession numbers, a more standard and universal identifier than b-numbers, will be added in the very near future. Novel proteins and genes are also supported by adding them directly via the administration interface.

Data output

EDD provides access to all the data pertaining to an experiment in the form of standardized output files and a RESTful API in order to access data programmatically (in development). See Screencasts 3 and 4 in Supporting Information for a demonstration.

Two output formats are provided at this time: comma separated values (CSV) and SBML. The CSV format is a general spreadsheet format providing selected information for a given experiment. Options on the CSV export can customize the output to include a subset of the data of interest. There are three basic options for spreadsheet layout (illustrated in Fig. S1 in Supporting Information):

- Rows of samples, columns of metadata and points; "short and wide". Suited for researchers reading data across lots of samples.
- Rows of data points, columns of metadata; "tall and skinny". Suited for loading into analysis packages like Spotfire or R.
- Rows of metadata and points, columns of samples; a transpose view of "short and wide". Suited for researchers reading lots of points across a few samples.

The SBML format is tailored to enable and facilitate flux analysis through COBRA methods[45] or ^{13}C MFA.[35] The SBML output contains exchange fluxes and growth rates calculated from the data stored in EDD as explained in the Supporting Information. In order to make the SBML output useful for ^{13}C MFA,[46] it was necessary to supplement the SBML standard with ways to include ^{13}C labeling patterns for different metabolites (see Supporting Information). New standards for different outputs can be added as explained in detail in the Supporting Information.

The RESTful API is structured along the hierarchies illustrated in Figs. 2 and 7 (see https://github.com/JBEI/edd/tree/master/docs/Interface.md). Accessing a Study will list all the Lines in the study, accessing a Line will list all the Assays on the line, and so on, until a script or program can access individual data points. When completed, the RESTful API will allow access to the data in EDD with more complex query criteria than a straightforward export can accommodate.

Read/edit permissions

EDD includes a permissions model for Studies. A Study will be created by default with only permissions for the creator to view or edit. Without adding alternate permissions, a Study will be private, visible only to the individual creating the Study. Additional permissions may be granted to individual users, to groups of users, or to all users with accounts on the EDD server. There are two types of permissions available: the Read permission allows for viewing, searching, and exporting data from a Study; and the Write permission allows for adding, modifying, importing, or deleting data from a Study, as well as modifying permissions on the Study.

Implementation

The EDD code is open source under a Berkeley Software Distribution (BSD) license. The front-end of EDD is written in TypeScript, JavaScript, and HTML/CSS. EDD runs in any modern web browser, but Chrome is recommended (https://www.google.com/chrome/). The back-end is coded in Python and built on the Django platform (see Fig. 6). The code and documentation are available on Github (https://github.com/JBEI/EDD) and is divided into the following modules:

Templates and Views

The Django template framework is used to handle the layout and structure of EDD pages. Templates enforce a separation between how data in EDD are processed and how the same data are presented. By separating processing and presentation, the code for both is easier to generalize and re-use. A base template defines the overall look-and-feel of application pages and consistent navigation across the application. Additional templates referencing the base template define the structure for the major pages within EDD (*e.g.*, show study details; or, import instrument data).

Individual requests to EDD are handled with view functions. EDD directs requests to

view functions based on the contents of the request URL. Then, the view function processes the data in the request, loads and updates data from the database, and builds a response using the view's template.

Front-end and visualization

The lines, bars, axes, and labels in the overview plot are rendered in SVG via the D3 JavaScript library (d3js.org). Hovering over any line or bar triggers a CSS-based visual effect to make it stand out from the others, and provides more details on the data behind the visualization.

The progressive filtering of metadata criteria is accomplished by creating a Typescript class for a filtering column that accepts and then emits a set of records, and then subclassing it for each of the base five kinds of metadata (Line, Strain, Protocol, Assay, Measurement), plus a sixth subclass for all the customized metadata types that can appear in a Study. The Measurement subclass is itself further subclassed for Metabolites, Proteins, and Transcripts. When a Study page loads, each of these classes is instantiated once, and the resulting filtering object is placed in an ordered list. Then, when a Study begins receiving data records from the server, additional instantiations of the customized metadata subclass are made, one for each new custom type detected. These objects are added to the beginning of the list.

Each object is responsible for a column in the filtering section, and for accumulating and then managing its list of unique values. To achieve progressive filtering, a set of all the data records in the Study is fed into the first object in the list, which then emits another set, possibly shortened by removing all the records that do not match any checked values in the column. That set is passed to the next object, and further reduced, and so on, until the final set is fed into the overview plot for display.

Database

Access to the EDD database is provided through the Django Object Relational Manager (ORM, Fig. 7). The ORM offers an interface to interact with entities in the database directly with Python code. This abstraction layer allows for EDD code to generally work with higher-level concepts of Studies, Assays, or Measurements instead of the underlying data models (*i.e.*, no need for SQL queries). Code execution can be triggered upon specified events through signal handlers in the ORM system. For example, a signal handler is responsible for updating Study information in the search index whenever a Study changes.

The data model for EDD centers on a few abstract concepts, tied together into the nested hierarchy of Study, Line, Assay, Measurement (Figs. 2 and 7). EDDObject defines the base for these parts of EDD. Each EDDObject has a unique machine-readable identifier, a human-readable name and description, update history, comments, files, and arbitrary metadata. Metadata, in turn, is defined by a MetadataType object. Each metadata value on an EDDObject references a MetadataType, containing the information needed for other code to interpret the value.

As an example, a Line is an EDDObject that has metadata describing the conditions of a biological sample. The specific metadata types used are customizable for each Line. The metadata that needs to be captured will differ between an experiment concerning cultures grown in flasks, compared to an experiment concerning corn growing in a field. Some metadata values, like Strain, are in turn EDDObjects, containing additional metadata. Lines concerning strains link to corresponding strain entries in a strain repository, such as ICE.

The definitions of metadata are fully configurable, and can leverage existing specifications of metadata, such as those included in the DICOM-SB standard[27] or ISA-Tab.[16]

Importers and Exporters

EDD defines an interface for generalized import and export of data in various formats. There are two types of inputs: a protocol-specific input from a particular instrument (*e.g.*, HPLC

or transcriptomics data), and a general import for data types not otherwise covered. Import modules transform the data into structures of the EDD database (Fig. 4). Export modules do the reverse process transforming selected data from the EDD database into other useful output formats. These modules are the primary way to move data into and out of EDD. Structuring the code as modules interfacing to and from the EDD database allows for the input of complex workflows through the flexible combination of these modules. Hence, an experiment that produced HPLC, transcriptomics, and proteomics data can have its data introduced in EDD through a successive application of the respective modules (Fig. 3).

Services

EDD makes use of several open-source systems to provide services to the main application. Each service is run using Docker containers (www.docker.com), allowing for standard installation and deployment across servers. Installing and running a service only requires having a Docker host and the name of a service image. Docker handles downloading all the packages and code needed to run the service in an image. No separate installation is required, and most service images will have a reasonable default configuration included.

Code for EDD is itself collected into an image that will run in a Docker container. A Dockerfile included in the source code describes all the required setup and install for the core EDD service, and can be built into an image that is run just like any other service. Building this image once will allow the same image to be copied to any Docker host and launch a new instance.

A simple overview of the services driving EDD is included in (Fig. 8). All services are contained within the Docker Host. EDD connects to the Internet and outside world at two points: with the Nginx web server (www.nginx.com) to handle web requests, and with the Exim mail server (http://www.exim.org/) to send email notifications. Incoming requests to Nginx get routed to the core EDD image running a Django website in the Gunicorn WSGI application server, or to a backend file storage service. The core EDD service connects

to several other services to implement specific features. Text search and faceting uses a Solr document index service (lucene.apache.org/solr/). A Redis cache (redis.io) stores login session information and copies of the latest versions of static web resources like images and scripts. The core data model of EDD is implemented with a SQL schema running in a PostgreSQL service (www.postgresql.org). Any tasks that would take longer than the duration of a typical web request are handled by a Celery service (www.celeryproject.org) running a copy of the EDD Docker image. Communication between the EDD application service and the EDD worker service is mediated by a RabbitMQ message queue service (www.rabbitmq.com). Management of the message queue is handled by an optional Flower service, which can also be connected to the Nginx service to enable management of the task queue from outside of the Docker host.

This microservice architecture of the EDD application ecosystem is intended to simplify the process of expanding an installation of EDD. All of the services represented by rectangular boxes in Fig. 8 are stateless services, meaning capacity can be added by replacing the service box with a simple load balancer dividing the workload among multiple container copies. The three stateful services: Solr, Redis, and Postgres; represented by upright cylinders, all offer their own clustering solutions to scale beyond a single node. The file storage service, represented by an overturned cylinder, can use any standard data storage strategy; from local disks, to large RAID arrays, to large cloud storage providers like Amazon AWS S3 buckets.

Results and discussion

In this section, we present two example workflows that use experimental data contained within EDD to produce actionable items for metabolic engineering. Another possible use of EDD is to store synthetic biology parts characterization data, as is demonstrated by the public version of EDD (https://public-edd.jbei.org). This instance of EDD holds the data

for all the synthetic biology parts characterized in a recent publication concerning a Cas9-based toolkit for instituting genetic changes in *S. cerevisiae* to optimize heterologous gene expression.[47]

The first workflow will show how to upload time-resolved HPLC data into EDD. We will demonstrate the visualization capabilities and then download the data as a SBML file. We will then show how to use this SBML file in conjunction with the COBRApy[31] library to predict intracellular metabolic fluxes (which provide a comprehensive description of cellular metabolism) through FBA (Flux Balance Analysis). FBA has important applications in bioengineering,[48,49] microbial ecology[50] and biomedicine.[51]

The second workflow will show how to upload targeted proteomics data into EDD, how to view these data and how to download them for further analysis. We provide an example of this further analysis by using the proteomics data obtained from a bioengineered *E. coli* strain to increase production of limonene, repeating an analysis done in a previous publication.[52]

Both of these workflows (and their input files) are demonstrated through Screencasts 4 and 5 in the Supporting Information, or at https://public-edd.jbei.org/pages/tutorials/.

Using metabolite concentration data to derive internal metabolic fluxes through Flux Balance Analysis (FBA)

This workflow demonstrates how to upload time-resolved HPLC data into EDD, visualize them and download them in the SBML format so internal metabolic fluxes can be calculated through FBA.[53] The full workflow is showcased in Screencast 4. We will first introduce the data in EDD in two steps (Fig. 3).

We start at the main page and click on "Add New Study" on the upper right. The initial step involves providing basic metadata information such as the study name, a brief description of the study and a contact person. This action prompts for an experiment description, which can be introduced by dragging and dropping the file "FBA_Experiment_Description.xlsx" (available as Supporting information and https://public-edd.jbei.org/pages/tutorials/). This

excel file contains a description of the experimental design on the basis of lines, as well as the protocols applied and the corresponding assays (Fig. 2). Line information includes links to detailed strain and plasmids information in ICE, as well as carbon source and media. In this case, this minimal example describes two shaking flask cultures (line BW1 and ArcA) of *E. coli* for which HPLC measurements of glucose and acetate are available at times 0, 7.5, 9.5, 11, 13, 15, and 17 hours. This template can be modified as desired to describe different experiments. As soon as the experiment description is uploaded, the user can view the corresponding lines and other experimental details.

The next step is to upload data by clicking on "Import Data" on the upper right corner. This action takes us to a data import page where the desired input format (the general import in this case) and corresponding protocol ("HPLC" in this case) are chosen. The HPLC data can be found in the "FBA_HPLC.xlsx" file. Dragging and dropping this file in the import page will make EDD parse the data and show an initial visualization, where the user can discard undesired time points (e.g. having resulted from experimental mistakes). EDD automatically matches the metabolite names to the database of standard metabolite names included, and the user can correct this assignment if needed. Once "Submit Import" is pressed, the data are now available on the main page of EDD for visualization. OD data is uploaded in an analagous manner.

The filtering section below the data graph provides the means to only look at certain parts of the data set. For example, clicking on 'arcA' below 'Strain' only shows the HPLC data corresponding to the arcA strain. Clicking on 'D-Glucose' below 'Metabolite' only shows the HPLC data corresponding to the glucose measurement. Clicking on both, only shows the acetate curves for the arcA strain (see Screencast 2 in the Supporting Information).

Data can be downloaded in a standardized format for later analysis. In this case we will download them in the SBML format. Exchange fluxes are automatically calculated from the extracellular metabolite concentrations described in the HPLC data (see Supporting Information). This file can be obtained by clicking on 'BW1' line, then selecting "Export

Data" and then selecting "to SBML" and "Take Action". This procedure will take the user to an export page that will determine the export parameters. The first one is which genome-scale model to use as a base (*i.e.*, which genome-scale model to apply the previously calculated exchange fluxes to). We will choose the *E. coli* iJO1366 model in this case, for the sake of example. The second step will involve selecting which OD measurement values will be used to constrain the biomass (biomass is assumed to be proportional to OD through a constant value that is explicitly provided in this section and can be changed as needed). These values are already preselected, so we only need to check that they are not obviously wrong (*e.g.* set to zero). Step three involves pairing the calculated exchange fluxes with the corresponding reactions in the genome-scale model. Finally, we can download the SBML file for the desired time point by clicking on "Download SBML".

The final step involves using the COBRApy library[31] and the SBML file downloaded to predict internal metabolic flux profiles through Flux Balance Analysis.[53] We can predict fluxes in five lines of code:

```
import cobra
model=cobra.io.read_sbml_model('EciJR904at20hrs.xml')
solution=model.optimize()
solution.objective_value
solution.fluxes[['PGI','GND']]
```

which shows a value of predicted flux of 2.61 mmol/gdw/hr for PGI (glucose-6-phosphate isomerase) and 0.91 mmol/gdw/hr for GND (hosphogluconate dehydrogenase). This code along with the expected results are shown in Jupyter notebook A in the Supporting Information.

Using targeted proteomics data to improve biofuel production through Principal Component Analysis (PCAP)

This workflow shows how to use EDD and the Scikit-learn library to leverage targeted proteomics data to improve biofuel production (limonene) by bioengineered *E. coli*, as demonstrated in Alonso-Gutierrez *et al.*[52] This workflow is showcased in Screencast 5 in the Supporting Information.

This example provides a demonstration of how to add several types of data using the two step process in Fig. 3. The initial steps of how to create a study are the same as for the previous example, in terms of providing the basic metadata. The description of the experiment can be found in 'PCAP_Experiment_Description.xls': in this case there are thirty shake flask cultures (lines 2X-Mh to 2X-Hm) of *E. coli* for which targeted proteomics data samples are taken at 24 hrs. Dragging and dropping the file into the page obtained by clicking on "Add New Study" creates a new study reflecting all these details. The proteomics data can be found in the "PCAP_Proteomics.csv" file. We can add these data to the study by clicking on "Import data" and following the instructions in the input page as shown in the previous example. This example has two additional data types associated besides the targeted proteomics data: limonene production measured through GC-MS ("PCAP_GCMS.csv" file) and optical density measured through spectroscopy ("PCAP_OD.xlsx" file). Adding the limonene measurements is as straightforward as pressing again "Import data" and following the instructions in the input page. Adding the optical density data follows the same procedure.

EDD offers several ways to visualize the data we previously loaded. In this example, the line graphs displaying the dependence with time are of limited use, since all data are collected at a single time. By clicking on "Bar Graphs" at the top of the "Data" tab, we can see this data in bar form grouped by measurement, line or time, as indicated by the different buttons. Hovering over each bar or data point gives further information. As before, we can filter certain types of data by clicking on "Filtering" and using the ensuing menu.

By clicking on a line, protocol, or protein, we only see the data corresponding to that line, protocol, or protein. The assays applied to each line and the sampling times are available by clicking on the "Table" tab.

We will now download the data from EDD for further analysis using Principal Component Analysis of Proteomics (PCAP[52]). First, we select the lines we would like to download and we click on "Export Data" and select "as CSV/etc" from the download menu options. This provides a CSV file with a defined format that can be used as input for Jupyter notebook B (see Supporting Information).

The next steps involve taking the proteomics and production data and use Principal Component Analysis to find which proteins need to have their expression changed in order to improve biofuel production. This procedure is carried out using the Scikit-learn library,[32] and is demonstrated in Jupyter notebook B. The input is the CSV file obtained from EDD, and the output is Fig. 4 from Alonso-Gutierrez *et al*,[52] which predicts which part of the proteomics phase space is associated to improved limonene production (see publication for further details).

Conclusion

We have presented in this manuscript EDD, an interactive online open-source tool that serves as a repository of experimental data. Linked with ICE, EDD provides a standardized description of experiments: from the strains and plasmids involved, to the protocols used, the experimental design for sampling, and the data extracted. While the initial use cases and the examples provided here are geared towards microorganism cultivation and phenotyping, the data schema and different functionalities can be adapted to other uses (*e.g.*, enzyme characterization or plant bioengineering).

Data input can be done either manually through a web interface or through automated workflows for typical data types. The latter includes input for: HPLC data, transcriptomics,

proteomics data, metabolomics data, and Biolector data. These workflows provide a drag-and-drop interface that parses data into the database automatically. These workflows are modular, and new modules can be written for additional data types (*e.g.*, chip-Seq, etc). Once the API in development is finished, it will provide the possibility of automating data input, and hence ease the integration of data from other databases and publications.

Data visualization is provided for each study through an interactive window where different data types can be seen simultaneously (Fig. 5). Different data types and strains can be interactively filtered in or out to facilitate comparisons. Data for each protocol can be found at the bottom of each study, along with sampling details.

Data standardization is enabled by forcing all data into an ontology and using standardized ontologies for data (for example, all metabolomics data uses the same metabolite names). Furthermore, the user is forced to include a minimum of metadata as a description of metadata. A flexible use of metadata means that, beyond that minimum obligatory core, extra metadata can be included, if desired, by the experimentalist.

Data output can be done using a variety of formats, including CSV or SBML files. These output streams are modular and new modules can be added for different output formats. By virtue of the internal organization of EDD, all data output is consistent and can be used to feed a variety of modeling or data mining approaches.

EDD improves on single -omics type databases such as PRIDE,[54] MOPED[55] and PAXdb[56] (for e.g. proteomics) because it is able to integrate multiple types of -omics data (e.g. transcriptomics, proteomics and metabolomics). Furthermore, the metadata typically stored in these systems (e.g. PRIDE) focuses on data acquisition and sample preparation metadata (i.e. trypsin amount, digestion length..), whereas experiment metadata (e.g shaking speed, culture volume, growth temperature) is typically lacking in these databases but is captured on EDD. However, while some of these databases provide data analysis capabilities (e.g. MOPED or PaxDb), EDD was not meant to perform complex data analysis. There are many available tools available for data analysis (e.g. through Kbase or Jupyter notebooks)

and we believe EDD's mission is not to choose those tools for the user but, rather, feed those tools the standardized data they need, in order to streamline their use (see for example the multi-omics data viewer Arrowland, https://public-arrowland.jbei.org/).

In this manuscript, we have described two use cases for EDD in metabolic engineering (all data available in the Supporting Information and https://public-edd.jbei.org/pages/tutorials/): 1) using extracellular metabolite concentrations to predict internal metabolic fluxes for an *E. coli* strain using FBA, and 2) using proteomics data to increase biofuel production in a bioengineered strain. These use cases are presented as tutorials and showcase the utility of EDD for metabolic engineering and synthetic biology applications. EDD is, however, a tool in continuous development. We present here a tool that addresses some of our current needs, but the code is available to be modified and adapted to fit other future needs that require collection and storage of large amounts of experimental data.

EDD also provides a platform to disseminate the data produced at one institution to other institutions, hence becoming a repository of data of use for testing and parametrizing models. For example, JBEI's[57] public instance of EDD (https://public-edd.jbei.org) holds the information for all the synthetic biology parts characterized in a recent JBEI publication which provides the largest, most comprehensive Cas9-based toolkit to quickly institute genetic changes in *S. cerevisiae* to optimize heterologous gene expression.[47] We expect to continue to seed JBEI's public instance of EDD with data related to future publications from LBNL (e.g. associated to JBEI or the Agile BioFoundry: http://agilebio.lbl.gov/), and very soon open the possibility to other external researchers of uploading their own data. An alternative is for external researchers to set their own instances of EDD (as explained in detail in the github repository, https://github.com/JBEI/edd/blob/master/docs/Developer_Setup.md). We also welcome contributions and joint development (see https://github.com/JBEI/edd/blob/master/Con to fit other user's needs. Our final goal is to create a web of EDDs for different institutions able to efficiently exchange data, as is the case for the web of registries (https://www.jbei.org/jbeis-inventory-of-composable-elements-ice-tutorial-now-available/).

In the current world, where there is an increasingly strong trend to disclose algorithms as open source code,[58] but training data is viewed as extremely valuable,[59] EDD will provide significant value as more experiments are available. We hope EDD will help enabling reproducibility and predictability in the fields of metabolic engineering and synthetic biology.

Competing interests

The authors declare that they have no competing interests.

Author's contributions

HGM, GWB, WCM, MF, NJH, TL designed the software. JAG and KWB helped conceive the tool functionalities. HGM, WCM, PDA, JDK, NJH, IV, EEKB, CP, ZC, DA, GWB, TWHB, JAG, KWB, and AM wrote the paper. WCM, GWB, MF, TL, TWHB, MD wrote the code.

Acknowledgement

This work was part of the DOE Joint BioEnergy Institute (http:// www.jbei.org) and part of the Agile BioFoundry (http://agilebiofoundry.org) supported by the U.S. Department of Energy, Energy Efficiency and Renewable Energy, Bioenergy Technologies Office, through contract DE-AC02-05CH11231 between Lawrence Berkeley National Laboratory and the U. S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access

Plan (http://energy.gov/downloads/doe-public-access-plan). NJH was also supported by the DOE Joint Genome Institute (https://jgi.doe.gov) by the U.S. Department of Energy, Office of Science, Office of Biological and Environmental Research, through contract DE-AC02-05CH11231 between Lawrence Berkeley National Laboratory and the U.S. Department of Energy. This research is also supported by the Basque Government through the BERC 2014-2017 program and by Spanish Ministry of Economy and Competitiveness MINECO: BCAM Severo Ochoa excellence accreditation SEV-2013-0323.

We acknowlege and thank Daniel Lopez for contributing EDD's logo. We also thank Jacob Coble, Sarah LaFrance, Xianwei "John" Meng, Oge Nnadi, Hector Plahar, Lisa Simirenko and Ernst Oberortner for reviewing EDD's source code and their helpful suggestions.

Supporting Information Available

The following supporting information is available:

- Supporting text including (SupportingText.pdf):
  - A detailed explanation of how exchange fluxes are calculcated for export from EDD using SBML.
  - An explanation of the extensions of SBML that were required in order to store ^{13}C labeling data, transcriptomics, proteomics, metabolomics and fluxomics data.
  - Instructions on how to add new output standards for EDD.
  - A supplementary figure explaning the different layouts of exported spreadsheets.
- Five screencasts as tutorials that show five fundamental functionalities in EDD:
  - Screencast 1: Data upload (1-Data Input.mp4).
  - Screencast 2: Data visualization (2-Data Visualization.mp4).
  - Screencast 3: Data download (3-Data Download.mp4).

- Screencast 4: Using metabolite data for flux analysis (4-FBA.mp4).
- Screencast 5: Using proteomics data to increase biofuel production (5-PCAP.mp4).

- A zip file (Examples.zip) containing Jupyter notebooks and input files to recreate:
  - Using metabolite data for flux analysis (Screencast 4):
    * Jupyter Notebook A.ipynb (+ corresponding html version)
    * FBA_Experiment_Description.xlsx
    * FBA_HPLC.xlsx
    * FBA_OD.xlsx.
  - Using proteomics data to increase biofuel production (Screencast 5):
    * Jupyter Notebook B.ipynb (+ corresponding html version)
    * PCAP_Experiment_Description.xlsx
    * PCAP_GCMS.csv
    * PCAP_OD. xlsx
    * PCAP_Proteomics.csv

References

(1) Russo, E. (2003) Special Report: The birth of biotechnology. *Nature 421*, 456–457.

(2) *Industrialization of Biology*; The National Academies Press, 2015.

(3) House, T. W. (2012) National Bioeconomy Blueprint April 2012. *Industrial Biotechnology 8*, 97–102.

(4) Tang, N., Ma, S., and Tian, J. *Synthetic Biology*; Elsevier BV, 2013; pp 3–21.

(5) Doudna, J. A., and Charpentier, E. (2014) The new frontier of genome engineering with CRISPR-Cas9. *Science 346*, 1258096.

(6) Gardner, T. S. (2013) Synthetic biology: from hype to impact. *Trends Biotechnol. 31*, 123–125.

(7) Chubukov, V., Mukhopadhyay, A., Petzold, C. J., Keasling, J. D., and Martín, H. G. (2016) Synthetic and systems biology for microbial production of commodity chemicals. *NPJ Syst. Biol. Appl. 2*, 16009.

(8) Prinz, F., Schlange, T., and Asadullah, K. (2011) Believe it or not: how much can we rely on published data on potential drug targets? *Nat. Rev. Drug Discovery 10*, 712–712.

(9) Begley, C. G., and Ellis, L. M. (2012) Drug development: Raise standards for preclinical cancer research. *Nature 483*, 531–533.

(10) Baker, M. (2016) 1,500 scientists lift the lid on reproducibility. *Nature 533*, 452–454.

(11) Karr, J. R., Sanghvi, J. C., Macklin, D. N., Gutschow, M. V., Jacobs, J. M., Bolival, B., Assad-Garcia, N., Glass, J. I., and Covert, M. W. (2012) A Whole-Cell Computational Model Predicts Phenotype from Genotype. *Cell 150*, 389–401.

(12) Hyduke, D. R., Lewis, N. E., and Palsson, B. Ø. (2013) Analysis of omics data with genome-scale models of metabolism. *Mol. BioSyst. 9*, 167–174.

(13) Nelli, F. *Python Data Analytics*; Springer Science Business Media, 2015; pp 237–264.

(14) Gill, R. T., Halweg-Edwards, A. L., Clauset, A., and Way, S. F. (2015) Synthesis aided design: The biological design-build-test engineering paradigm? *Biotechnol. Bioeng. 113*, 7–10.

(15) Davidsohn, N., Beal, J., Kiani, S., Adler, A., Yaman, F., Li, Y., Xie, Z., and Weiss, R. (2015) Accurate Predictions of Genetic Circuit Behavior from Part Characterization and Modular Composition. *ACS Synth. Biol. 4*, 673–681.

(16) Rocca-Serra, P., Brandizi, M., Maguire, E., Sklyar, N., Taylor, C., Begley, K., Field, D., Harris, S., Hide, W., Hofmann, O., Neumann, S., Sterk, P., Tong, W., and Sansone, S.-A. (2010) ISA software suite: supporting standards-compliant experimental annotation and enabling curation at the community level. *Bioinformatics 26*, 2354–2356.

(17) Brazma, A. et al. (2001) Minimum information about a microarray experiment (MIAME)-toward standards for microarray data. *Nat Genet 29*, 365–71.

(18) Taylor, C. F. (2006) Minimum Reporting Requirements for Proteomics: A MIAPE Primer. *PROTEOMICS 6*, 39–44.

(19) Clough, E., and Barrett, T. *Methods in Molecular Biology*; Springer Science Business Media, 2016; pp 93–110.

(20) Brazma, A. (2003) ArrayExpress–a public repository for microarray gene expression data at the EBI. *Nucleic Acids Res. 31*, 68–71.

(21) Jones, P. (2006) PRIDE: a public repository of protein and peptide identifications for the proteomics community. *Nucleic Acids Res. 34*, D659–D663.

(22) Vizcaíno, J. A. et al. (2014) ProteomeXchange provides globally coordinated proteomics data submission and dissemination. *Nat Biotechnol 32*, 223–226.

(23) Desiere, F. (2006) The PeptideAtlas project. *Nucleic Acids Res. 34*, D655–D658.

(24) Haug, K., Salek, R. M., Conesa, P., Hastings, J., de Matos, P., Rijnbeek, M., Mahendraker, T., Williams, M., Neumann, S., Rocca-Serra, P., Maguire, E., Gonzalez-Beltran, A., Sansone, S.-A., Griffin, J. L., and Steinbeck, C. (2012) MetaboLights–an open-access general-purpose repository for metabolomics studies and associated metadata. *Nucleic Acids Res. 41*, D781–D786.

(25) Gonzalez-Beltran, A., Maguire, E., Georgiou, P., Sansone, S.-A., and Rocca-Serra, P.

(2013) Bio-GraphIIn: a graph-based integrative and semantically-enabled repository for life science experimental data. *EMBnet.journal 19*, 46.

(26) Pianykh, O. S. *Digital Imaging and Communications in Medicine (DICOM)*; Springer Nature, 2011; pp 3–5.

(27) de Murieta, I. S., Bultelle, M., and Kitney, R. I. (2016) Toward the First Data Acquisition Standard in Synthetic Biology. *ACS Synth. Biol.*

(28) Finney, A., and Hucka, M. (2003) Systems biology markup language: Level 2 and beyond. *Biochm. Soc. Trans. 31*, 1472–1473.

(29) Hucka, M. *Encyclopedia of Systems Biology*; Springer Science Business Media, 2013; pp 2057–2063.

(30) 2015 Data Science Report. 2015; `https://visit.crowdflower.com/2015-data-scientist-report`.

(31) Ebrahim, A., Lerman, J. A., Palsson, B. O., and Hyduke, D. R. (2013) COBRApy: COnstraints-Based Reconstruction and Analysis for Python. *BMC Syst. Biol. 7*, 74.

(32) others,, et al. (2011) Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res. 12*, 2825–2830.

(33) Ham, T. S., Dmytriv, Z., Plahar, H., Chen, J., Hillson, N. J., and Keasling, J. D. (2012) Design implementation and practice of JBEI-ICE: an open source biological part registry platform and tools. *Nucleic Acids Res. 40*, e141–e141.

(34) Wiechert, W. (2001) 13C Metabolic Flux Analysis. *Metab. Eng. 3*, 195–206.

(35) Martín, H. G., Kumar, V. S., Weaver, D., Ghosh, A., Chubukov, V., Mukhopadhyay, A., Arkin, A., and Keasling, J. D. (2015) A Method to Constrain Genome-Scale Models with 13C Labeling Data. *PLoS Comput. Biol. 11*, e1004363.

(36) Funke, M., Buchenauer, A., Schnakenberg, U., Mokwa, W., Diederichs, S., Mertens, A., Müller, C., Kensy, F., and Büchs, J. (2010) Microfluidic biolector-microfluidic bioprocess control in microtiter plates. *Biotechnol. Bioeng. 107*, 497–505.

(37) Wilkins, C. L. (2000) Books and Software: Data mining with Spotfire Pro 4.0. *Anal. Chem. 72*, 550 A–550 A.

(38) Bolton, E. E., Wang, Y., Thiessen, P. A., and Bryant, S. H. (2008) PubChem: integrated platform of small molecules and biological activities. *Annu. Rep. Comput. Chem. 4*, 217–241.

(39) Schellenberger, J., Park, J. O., Conrad, T. M., and Palsson, B. Ø. (2010) BiGG: a Biochemical Genetic and Genomic knowledgebase of large scale metabolic reconstructions. *BMC bioinf. 11*, 213.

(40) Degtyarenko, K., De Matos, P., Ennis, M., Hastings, J., Zbinden, M., McNaught, A., Alcántara, R., Darsow, M., Guedj, M., and Ashburner, M. (2007) ChEBI: a database and ontology for chemical entities of biological interest. *Nucleic Acids Res. 36*, D344–D350.

(41) Weininger, D., Weininger, A., and Weininger, J. L. (1989) SMILES. 2. Algorithm for generation of unique SMILES notation. *J. Chem. Inf. Model. 29*, 97–101.

(42) Consortium, U. (2015) UniProt: a hub for protein information. *Nucleic Acids Res. 43*, D204–12.

(43) Blattner, F. R., Plunkett, G., Bloch, C. A., Perna, N. T., Burland, V., Riley, M., Collado-Vides, J., Glasner, J. D., Rode, C. K., and Mayhew, G. F. (1997) The complete genome sequence of Escherichia coli K-12. *Science 277*, 1453–1462.

(44) Benson, D. A., Cavanaugh, M., Clark, K., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., and Sayers, E. W. (2012) GenBank. *Nucleic Acids Res. 41*, D36–D42.

(45) Schellenberger, J., Que, R., Fleming, R. M. T., Thiele, I., Orth, J. D., Feist, A. M., Zielinski, D. C., Bordbar, A., Lewis, N. E., Rahmanian, S., Kang, J., Hyduke, D. R., and Palsson, B. Ø. (2011) Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox v2.0. *Nat Protoc 6*, 1290–1307.

(46) Martín, H., Kumar, V., Weaver, D., Ghosh, A., Chubukov, V., Mukhopadhyay, A., Arkin, A., and Keasling, J. (2015) A Method to Constrain Genome-Scale Models with 13C Labeling Data. *PLoS Comput Biol 11*, e1004363.

(47) Apel, A. R., d Espaux, L., Wehrs, M., Sachs, D., Li, R. A., Tong, G. J., Garber, M., Nnadi, O., Zhuang, W., Hillson, N. J., Keasling, J. D., and Mukhopadhyay, A. (2016) A Cas9-based toolkit to program gene expression in Saccharomyces cerevisiae. *Nucleic Acids Res. 45*, 496–508.

(48) Yim, H. et al. (2011) Metabolic engineering of Escherichia coli for direct production of 1,4-butanediol. *Nat. Chem. Biol. 7*, 445–452.

(49) Park, J. H., Lee, K. H., Kim, T. Y., and Lee, S. Y. (2007) Metabolic engineering of Escherichia coli for the production of L-valine based on transcriptome analysis and in silico gene knockout simulation. *Proc. Natl. Acad. Sci. U. S. A. 104*, 7797–7802.

(50) Stolyar, S., Dien, S. V., Hillesland, K. L., Pinel, N., Lie, T. J., Leigh, J. A., and Stahl, D. A. (2007) Metabolic modeling of a mutualistic microbial community. *Mol. Syst. Biol. 3*.

(51) Frezza, C. et al. (2011) Haem oxygenase is synthetically lethal with the tumour suppressor fumarate hydratase. *Nature 477*, 225–228.

(52) Alonso-Gutierrez, J., Kim, E.-M., Batth, T. S., Cho, N., Hu, Q., Chan, L. J. G., Petzold, C. J., Hillson, N. J., Adams, P. D., Keasling, J. D., Martin, H. G., and Lee, T. S. (2015) Principal component analysis of proteomics (PCAP) as a tool to direct metabolic engineering. *Metabol. Eng. 28*, 123–133.

(53) Lewis, N. E., Nagarajan, H., and Palsson, B. O. (2012) Constraining the metabolic genotype–phenotype relationship using a phylogeny of in silico methods. *Nat. Rev. Microbiol.*

(54) Vizcaíno, J. A., Csordas, A., Del-Toro, N., Dianes, J. A., Griss, J., Lavidas, I., Mayer, G., Perez-Riverol, Y., Reisinger, F., and Ternent, T. (2015) 2016 update of the PRIDE database and its related tools. *Nucleic Acids Res. 44*, D447–D456.

(55) Kolker, E., Higdon, R., Haynes, W., Welch, D., Broomall, W., Lancet, D., Stanberry, L., and Kolker, N. (2011) MOPED: model organism protein expression database. *Nucleic Acids Res. 40*, D1093–D1099.

(56) Wang, M., Weiss, M., Simonovic, M., Haertinger, G., Schrimpf, S. P., Hengartner, M. O., and von Mering, C. (2012) PaxDb, a database of protein abundance averages across all three domains of life. *Mol. Cell. Proteomics 11*, 492–500.

(57) Scheller, H. V., Singh, S., Blanch, H., and Keasling, J. D. (2010) The Joint BioEnergy Institute (JBEI): Developing New Biofuels by Overcoming Biomass Recalcitrance. *BioEnergy Res. 3*, 105–107.

(58) Metz, C. Google Just Open Sourced TensorFlow, Its Artificial Intelligence Engine. 2015; `https://www.wired.com/2015/11/google-open-sources-its-artificial-intelligence-engine/`.

(59) Vanian, J. IBM bought The Weather Company because weather affects nearly everything. 2015; `http://fortune.com/2015/10/28/ibm-weather-company-acquisition-data/`.

Figures

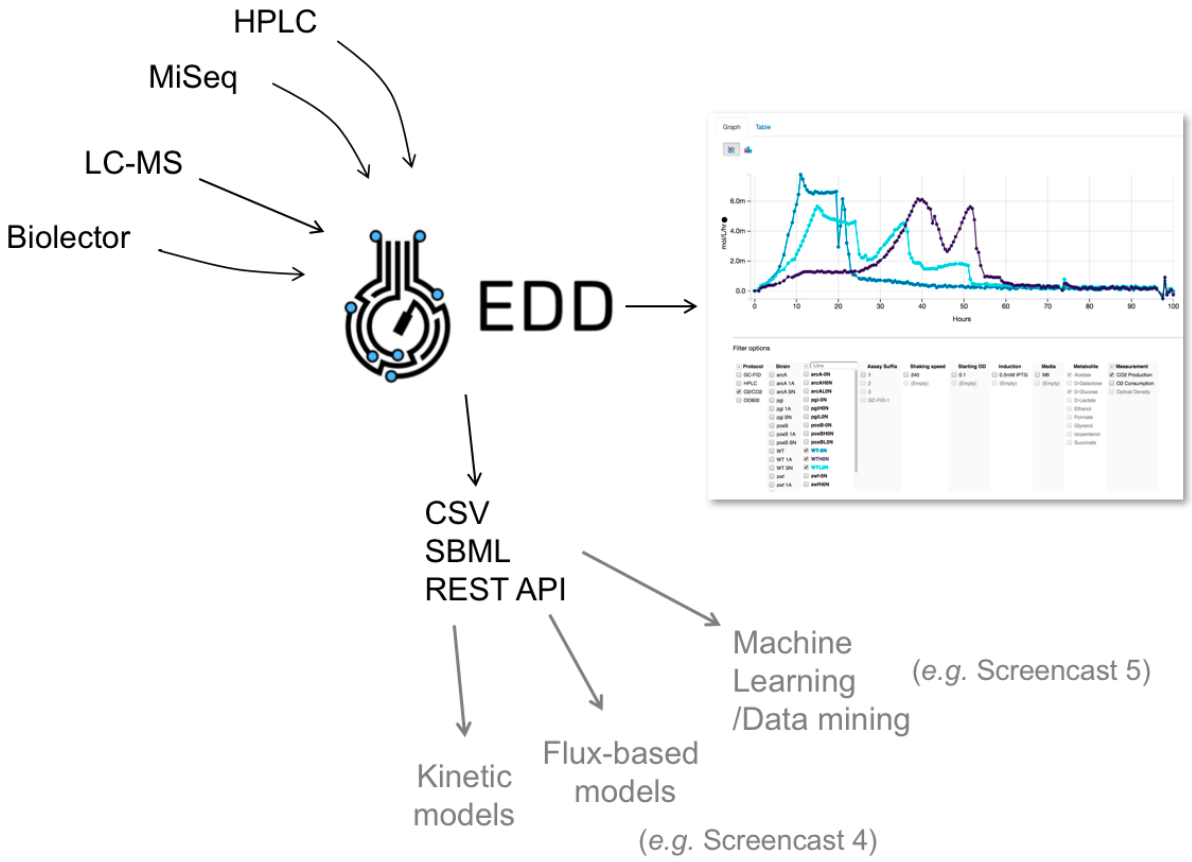

Figure 1: **Overview and key capabilites of EDD.** EDD collects data from different instruments, stores and visualizes them in an interactive way, and enables downloading them in a standardized format for use with a variety of modeling and analysis techniques. Screencasts 4 and 5, available in the Supporting Information (or https://public-edd.jbei.org/pages/tutorials/), provide step by step example tutorials to calculate internal metabolic fluxes, or to use proteomics data to improve biofuel production through data mining.

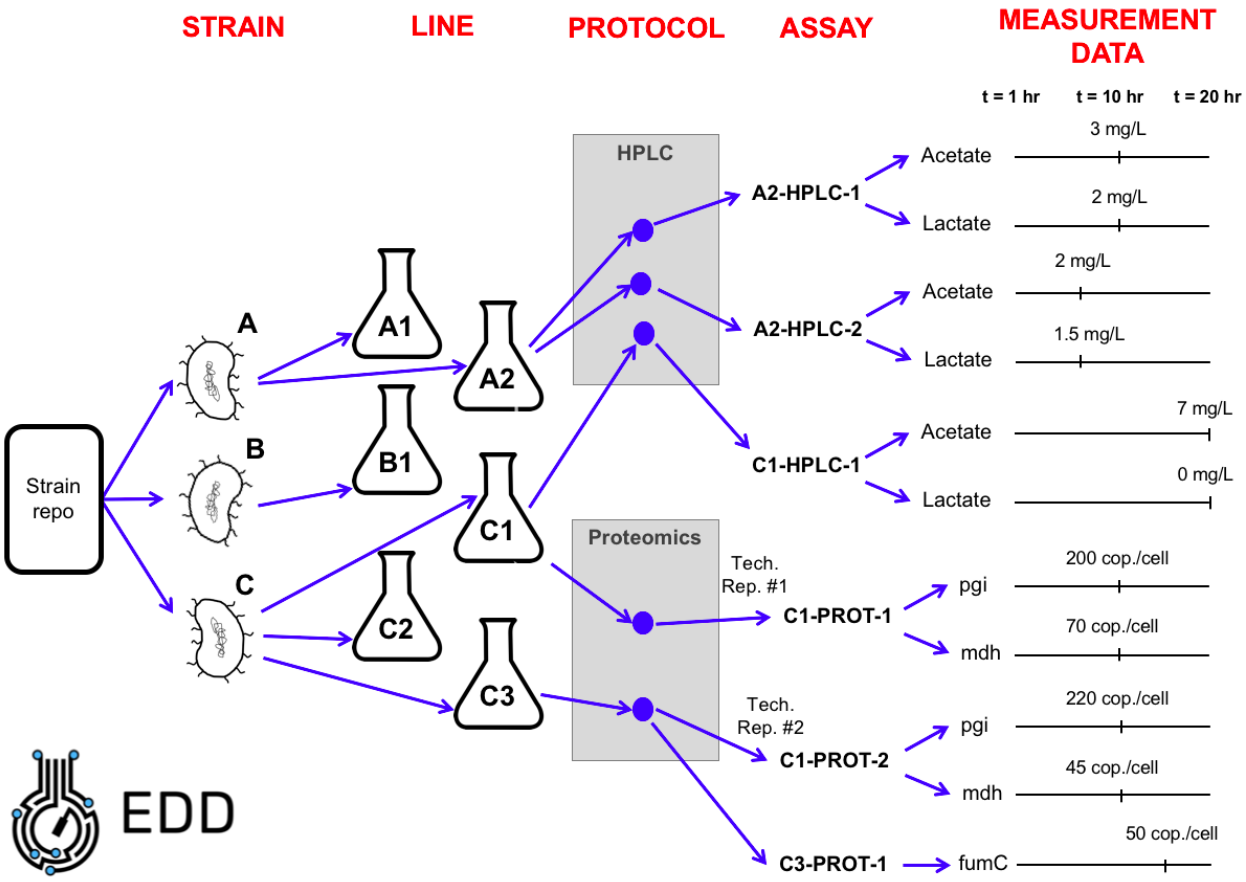

Figure 2: **Experiment description on EDD.** Example of how a common experiment would be described in EDD. This *study* involves culturing three *strains* (A, B and C) from a strain repository in several shaking flasks. Strain A is cultured in two flasks giving rise to two *lines* (A1 and A2). Strain B is cultured in a single flask (line B1) and strain C is cultured in three different flasks (lines C1, C2, and C3). The HPLC (High Pressure Liquid Chromatography measuring extracellular metabolite concentrations) *protocol* is applied to line A2 at t=10 hr giving rise to *assay* A2-HPLC-1. For assay A2-HPLC-1 the *measurement data* for acetate and lactate were 3 and 2 mg/L, respectively, at t=10 hr. Line C1 is subject to two different protocols: HPLC (t= 20 hr) and proteomics (quantitative measurement of expressed proteins, t=10 hr). Proteomics assay C1-PROT-1 on line C1 yields a measurement of 200 copies of pgi (phosphoglucose isomerase) per cell, and 70 copies of mdh (malate dehydrogenase) per cell at t=10 hr. A technical replicate of this measurement, coming from a different line (flask), constitutes a different assay C1-PROT-2.

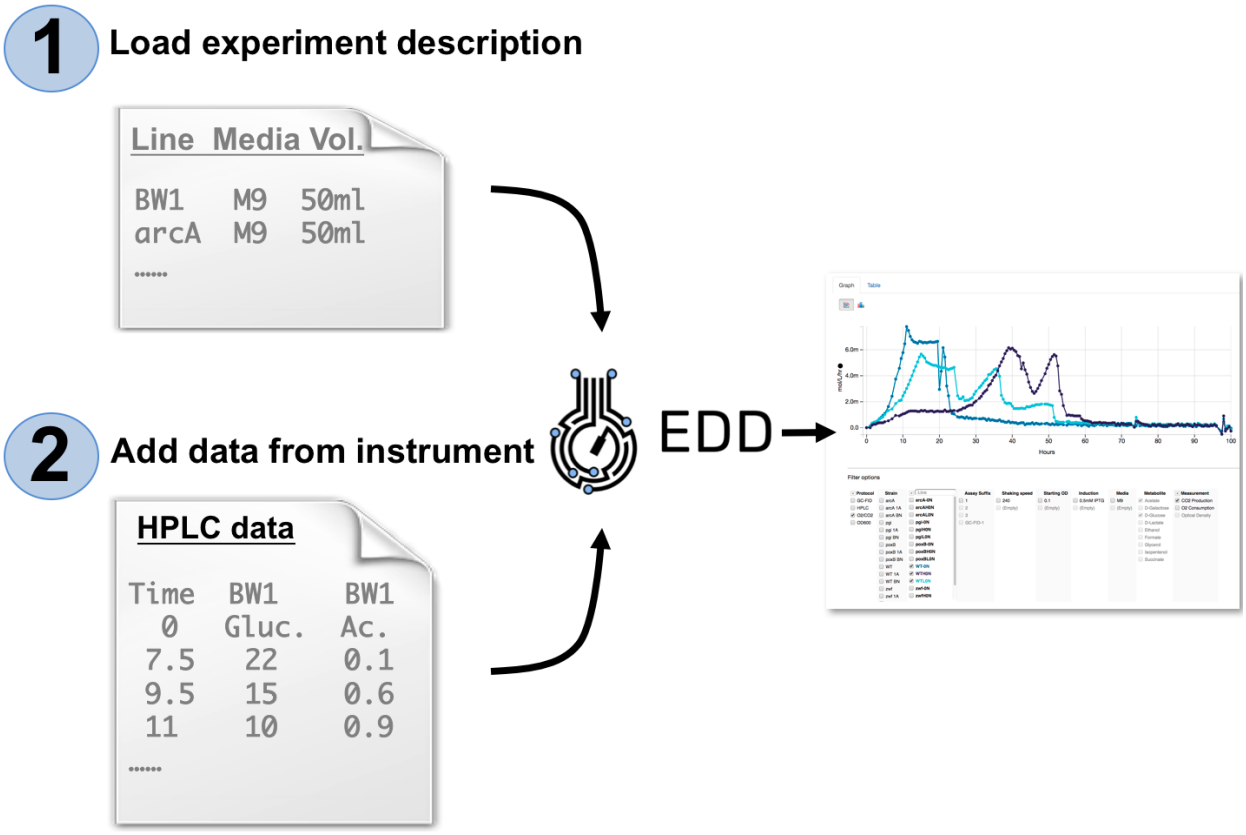

Figure 3: **Data input into EDD** has been streamlined. Users can input data in two steps. The first step involves adding description of the experiment describing lines and metadata for the study, as exemplified in Fig. 2. The second step involves uploading the data: (*e.g.*, HPLC data with metabolite concentrations). The input is modular, so additional data (*e.g.*, proteomics, transcriptomics, *etc.*) can be added later using the same import protocols. See Screencast 1 in the Supporting Information, or at https://public-edd.jbei.org/pages/tutorials/, for a demonstration.

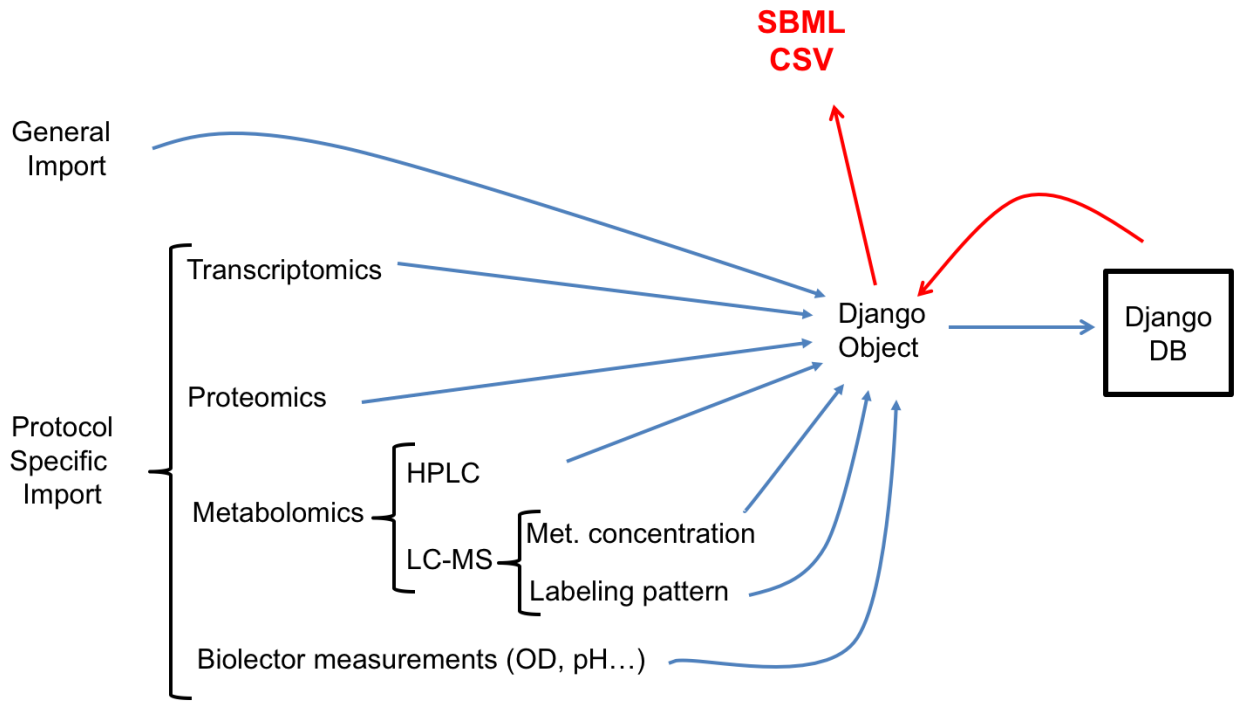

Figure 4: **Export and import modules.** Inputs are divided into two groups: protocol specific import that comes from a specific machine with a predetermined format, and a general import. Inputs are written so as to produce a Django object that is then stored in the database. The same modules are used for data export in SBML and CSV format. See Screencast 3 in the Supporting Information, or at https://public-edd.jbei.org/pages/tutorials/, for a demonstration of data export.

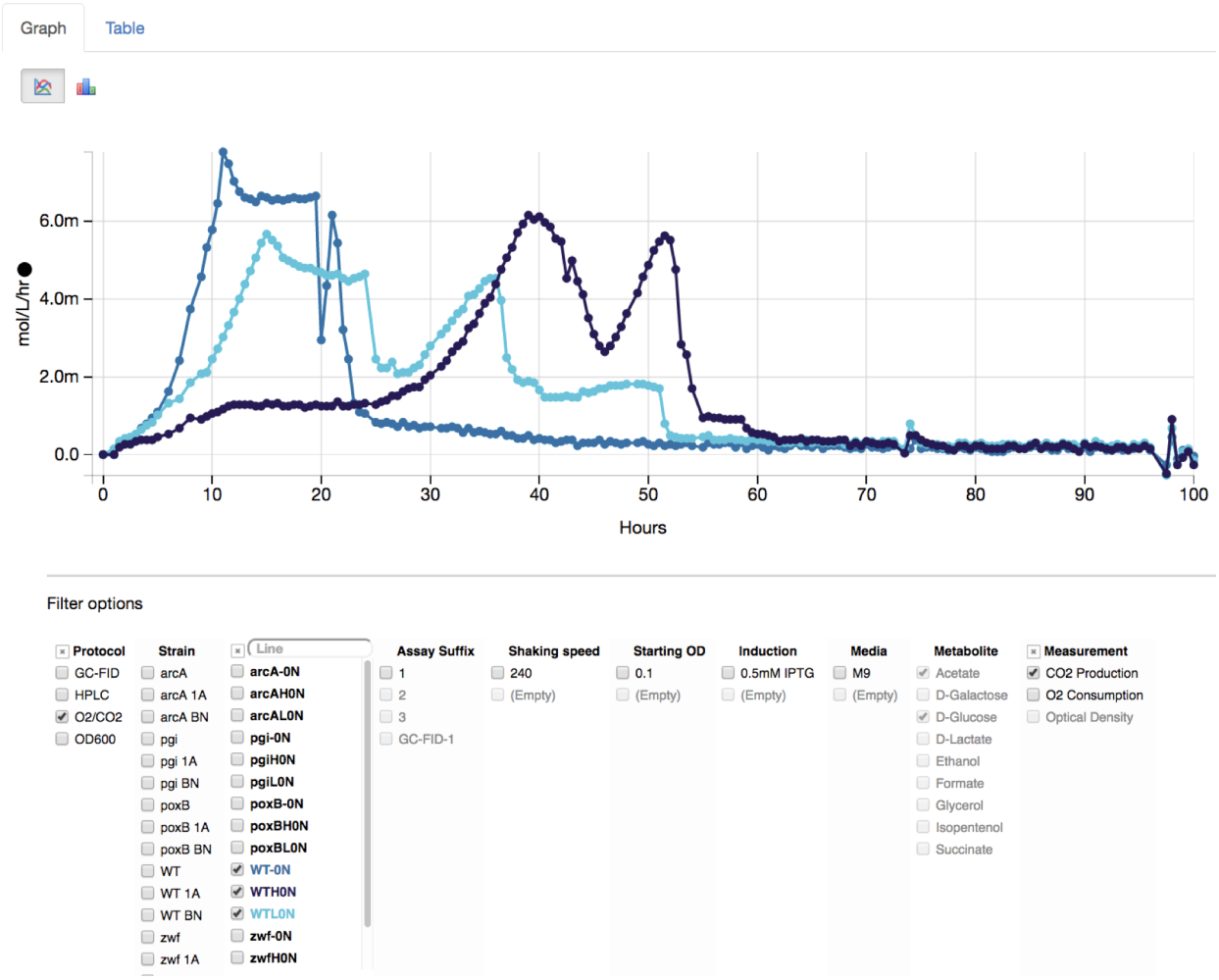

Figure 5: **Interactive data visualization.** The "Data" tab provides an interactive visualization of all data contained in a single study. The "Filter options" menu contains a classification of data and metadata. By clicking on each of the buttons in the menu one can choose to view *e.g.*, only the acetate, D-glucose, and O_2 consumption data for the 'WT BN' line. The user can also compare lines by checking them (*e.g.*, 'WT BN' vs 'WT 1A'). See Screencast 2 in the Supporting Information, or at https://public-edd.jbei.org/pages/tutorials/, for a demonstration.

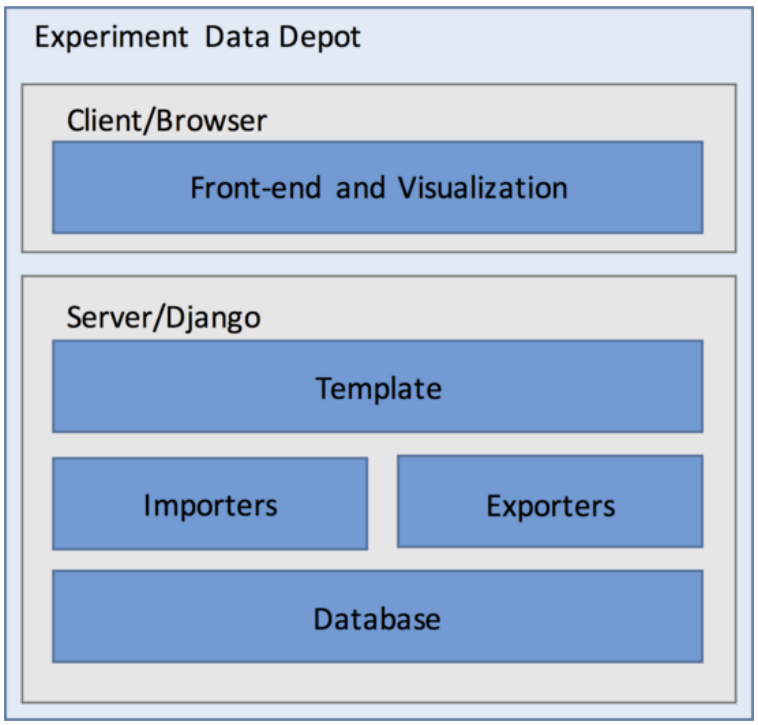

Figure 6: **High-level diagram of EDD code structure.** The front-end and visualization run on the client (internet browser) and are coded in TypeScript. The backend involves the database, importer, exporters and the templates and is coded in python using the Django framework.

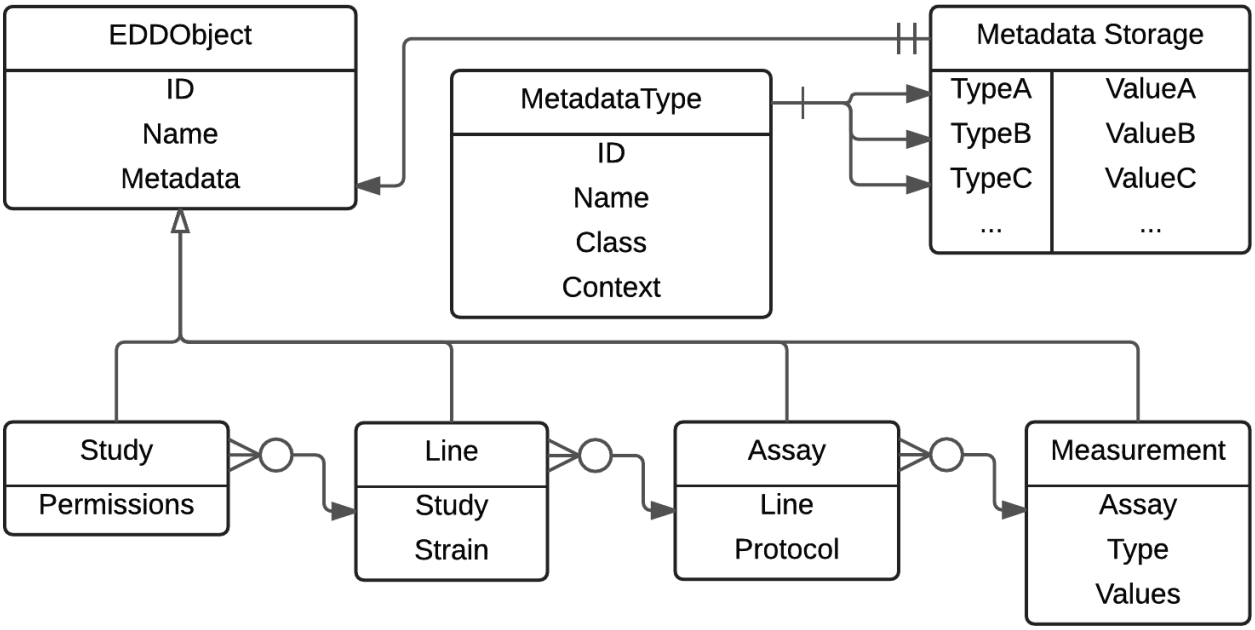

Figure 7: **Database schema for EDD data.** The database is accessed through the Django Object Relational Manager (ORM) and encodes the experiment descriptors shown in Fig. 2.

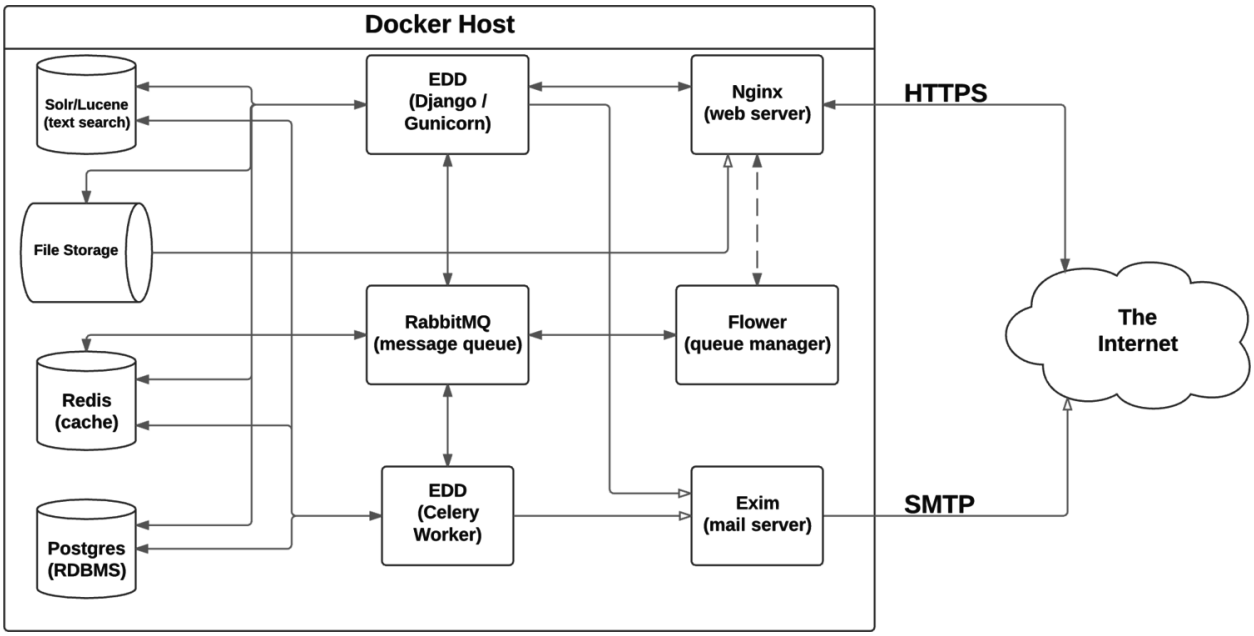

Figure 8: **Service diagram for EDD.** Multiple services combine together to create EDD. This microservice architecture simplifies the process of expanding an installation of EDD.

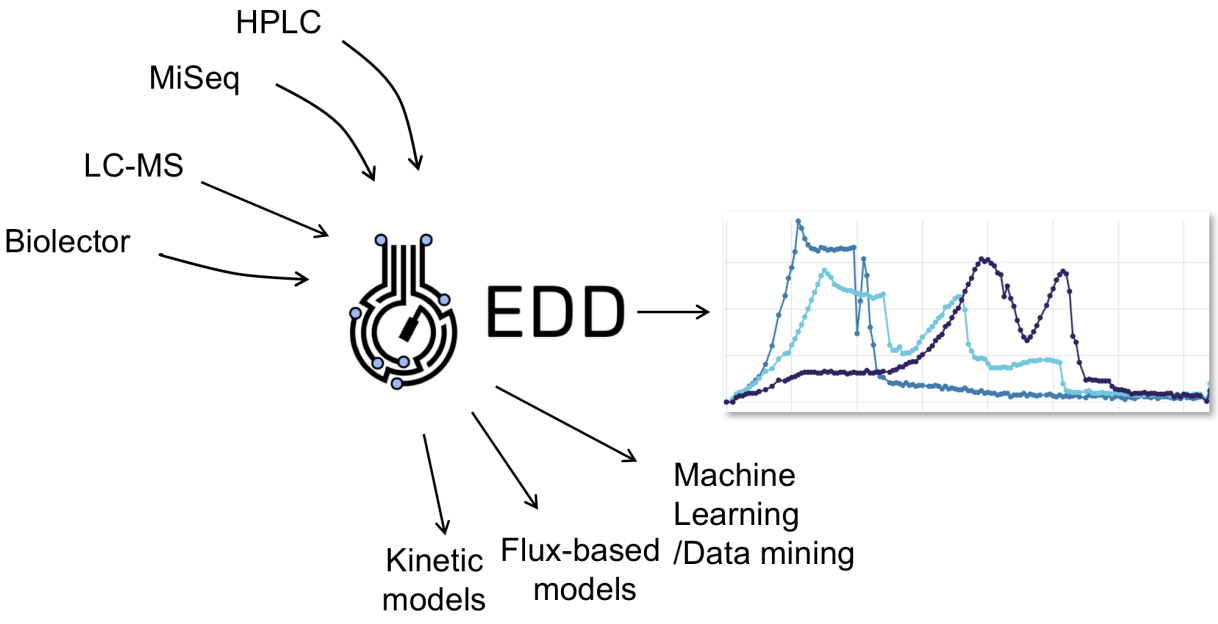

TABLE OF CONTENTS (ToC) graphic.